O Que é Polimorfismo ?

Significado da Palavra

O termo “polimorfismo” vem do grego e significa “muitas formas”. Em programação orientada a objetos (POO), polimorfismo permite que um mesmo método tenha comportamentos diferentes dependendo do tipo do objeto que o chama. Isso ajuda a criar programas mais flexíveis, reutilizáveis e fáceis de manter.

Tipos de Polimorfismo em Java

Java suporta dois tipos principais de polimorfismo:

1. Polimorfismo Estático (em tempo de compilação)

Esse tipo acontece quando o compilador consegue decidir qual método chamar antes do programa rodar. O exemplo clássico disso é sobrecarga de métodos (method overloading).

Exemplo:

```
public class Operacoes {
    public int somar(int a, int b) {
        return a + b;
    }

    public double somar(double a, double b) {
        return a + b;
    }
}
```

Aqui, o compilador escolhe qual método usar com base nos tipos dos parâmetros. Não há nenhuma decisão em tempo de execução — tudo é resolvido durante a compilação.

2. Polimorfismo Dinâmico (em tempo de execução)

Esse tipo ocorre quando a decisão de qual método executar só é tomada **durante a execução do programa**, com base no tipo real do objeto. Isso é feito através da **sobrescrita de métodos** (method overriding) em uma relação de herança.

Exemplo:

```
class Animal {
    public String emitirSom() {
        return "Som de animal";
    }
}

class Cachorro extends Animal {
    @Override
    public String emitirSom() {
        return "Au au!";
    }
}

class Gato extends Animal {
    @Override
    public String emitirSom() {
        return "Miau";
    }
}
 Animal a;

a = new Cachorro();
System.out.println(a.emitirSom()); // Saída: Au au!

a = new Gato();
System.out.println(a.emitirSom()); // Saída: Miau
```

Aqui, a variável a tem o tipo declarado Animal, mas o método que será chamado depende do **tipo real do objeto** (Cachorro ou Gato). A escolha do método certo é feita **em tempo de execução**, usando **tabelas de despacho dinâmico (vtable)**.

Como Funciona na JVM (Java Virtual Machine)

Quando o código Java é compilado, ele se transforma em **bytecode**, que é executado pela JVM. No bytecode, cada classe tem uma estrutura chamada **tabela de métodos virtuais (virtual method table ou vtable)**.

Funcionamento Simplificado:

1. O compilador gera bytecode e cria uma vtable para cada classe que tenha métodos sobrescritos.
2. Quando o programa chama obj.metodo(), a JVM consulta a vtable do objeto real (não da variável) para encontrar o método correto.
3. Isso é feito com ponteiros internos e índices. A JVM é otimizada para fazer isso rapidamente, mas ainda assim há um pequeno custo de performance em comparação com métodos estáticos.

Como o Sistema Operacional e o Processador Lidam com Isso

- **Sistema Operacional (SO)**: Não entende o que é polimorfismo. Ele apenas fornece recursos como memória, tempo de CPU e acesso ao disco para a JVM. O polimorfismo é gerenciado inteiramente pela JVM.
- **Processador**: Executa instruções de máquina convertidas a partir do bytecode. Ele também não "sabe" o que é polimorfismo — ele apenas segue as instruções que a JVM passa. A JVM traduz chamadas de métodos virtuais para instruções específicas que o processador consegue entender, geralmente envolvendo chamadas indiretas (por ponteiros).

Memória: Onde Fica Cada Coisa

1. **Métodos sobrescritos** ficam na **vtable** associada a cada classe.
2. Quando um objeto é criado no **heap**, ele carrega um ponteiro para a vtable da sua classe.
3. Ao chamar um método polimórfico, a JVM usa esse ponteiro para achar a versão correta do método.

Vantagens do Polimorfismo

- Reutilização de código: Um método pode trabalhar com diferentes tipos de objetos.
- Manutenção fácil: Mudanças em uma classe concreta não afetam quem usa a interface genérica.
- Flexibilidade: Permite escrever código mais genérico e independente de tipos específicos.

Desvantagens

- Performance: Chamadas polimórficas são ligeiramente mais lentas que chamadas diretas, por conta da verificação em tempo de execução.
- Complexidade: Pode deixar o código mais difícil de entender se usado em excesso, especialmente com muitos níveis de herança.
- Uso de memória: O uso de vtables e referências extras pode aumentar o consumo de memória.